(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 April 2002 (11.04.2002)

PCT

(10) International Publication Number
WO 02/29698 A2

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US01/42485

(22) International Filing Date: 5 October 2001 (05.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/238,283 5 October 2000 (05.10.2000) US
09/878,627 11 June 2001 (11.06.2001) US

(71) Applicant: **PROCURI.COM, INC.** [US/US]; 3348 Peachtree Street, N.E., Suite 200, Atlanta, GA 30326 (US).

(72) Inventors: **BROOKE, Steven R.**; 1200 Mayfield Manor Drive, Alpharetta, GA 30004 (US). **MCCLOSKEY, Michael A.**; 5910 Shadewater Drive, Cumming, GA 30041 (US).

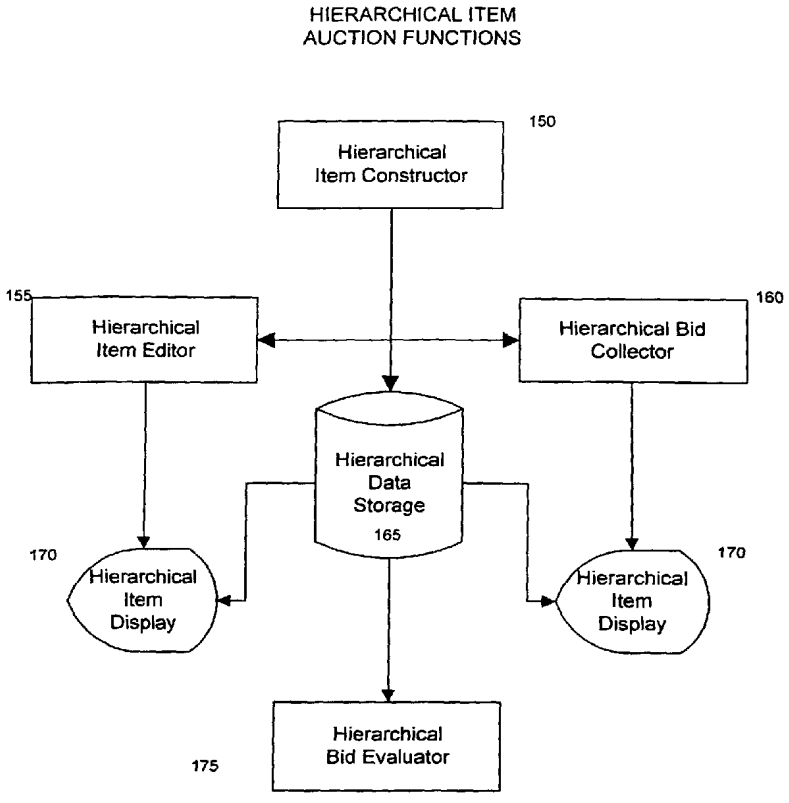
(74) Agent: **PETTY, W. Scott**; King & Spalding, 191 Peachtree Street, Atlanta, GA 30303-1763 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

*[Continued on next page]*

(54) Title: SYSTEM AND METHOD FOR HIERARCHICAL ADMINISTRATION OF COMPLEX ITEM STRUCTURES FOR ON-LINE AUCTION ENVIRONMENTS




(57) Abstract: A system implemented on a computer network for auctioning complex items to bidders. The system supports the definition and presentation of a complex item and its subassemblies in a distributed computing environment that facilitates the administration of an auction. An auction author defines the complex item to be auctioned by inputting information about the complex item and the subassemblies of which the complex item is comprised. The complex item and subassemblies can be presented to on-line bidders in a hierarchy format enabling bidding on individual subassemblies or an entire complex item. The system can evaluate competing bids from various bidders and select winning bids.

WO 02/29698 A2

**Published:**
— *without international search report and to be republished upon receipt of that report*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

# SYSTEM AND METHOD FOR HIERARCHICAL ADMINISTRATION OF COMPLEX ITEM STRUCTURES FOR ON-LINE AUCTION ENVIRONMENTS

## RELATED APPLICATION

This patent application claims priority, under 35 U.S.C. § 119, to U.S. Provisional Patent Application Serial No. 60/238,283, filed on October 5, 2000.

## TECHNICAL FIELD

The present invention is generally directed to the auction of items in a distributed computing environment. More specifically, the present invention provides a system and method for users of a distributed computing environment to easily create an auction for complex items consisting of multiple subassemblies.

## BACKGROUND OF THE INVENTION

The viability of on-line auction systems conducted over computer networks, such as the Internet, has been proven through the increasing consumer and business acceptance and participation in these systems. On-line auction systems can provide a mechanism for marshalling the auction process including activities for defining the item to be auctioned, posting the item description for review by all potential bidders, managing the bidding process via well established bid evaluation algorithms, selecting a winner based on the defined auction algorithm, and notifying the winner of the auction.

The success of an auction generally requires all participants to have a consistent understanding of the item(s) being auctioned. A consistent understanding enables participants to place a value on the item(s) with the same understanding of product requirements and features. This is easily accomplished for those items that are common or simple. If an item is composed of a single element or a few easily defined components, the chance of a successful auction is greatly increased.

Bid evaluation for an auction varies based on the selected algorithm. There are several types of bid evaluation algorithms currently in use. These common algorithms are English, Reverse, Dutch, Fixed-price, and Sealed-bid. An English auction is an ascending-price auction where bids must be higher in price than existing bids to win. A Reverse auction is the same as an English auction except that the auction is a descending-price auction where the lowest price

wins. A Dutch auction is technically an open-outcry, descending-price, multi-unit auction where bids are placed for partial quantity and the price decreases for the quantity remaining in the auction until all units are sold. A Fixed-price auction is one in which there is no bid increment. The price remains the same throughout the auction. A Sealed-bid auction is either an English or Reverse auction in which all bids are concealed.

The application of these common bid evaluation algorithms drives the auction process and is central to the winner selection process. Bid evaluation is straightforward for simple item structures. The evaluation of prices for these structures typically involves a simple price-price and time-time evaluation. There is no associative information that must be assessed during the evaluation process.

Existing on-line auction systems do not support the auctioning of complex items consisting of interrelated subassemblies. The conventional approach is one of simple price and time comparisons suited for individual items such as lumber or aluminum powder. However, contemporary business often involves complex items comprising multiple subassemblies, such as manufacturing equipment. Moreover, there are often complex relationships among the subassemblies comprising an item. A complex item structure has pricing information associated with each sub-assembly and sub-item, which may or may not be important to the price evaluation process. This additional information complicates the auction and increases the potential for misinterpretations that thwart a successful auction.

In view of the foregoing, there is a need in the art for a system which will support the auction of complex items. Specifically, a need exists in the art for a system that will allow for collecting and organizing of information about a complex item and its subassemblies that are to be auctioned. A further need exists to track the associative information with each subassembly comprising a complex item so that the subassemblies can be auctioned properly. There is also a need to present the information concerning the complex item and subassemblies to bidders in a fashion facilitating a successful auction. Finally, a further need exists to be able to receive and evaluate bids on the subassemblies comprising a complex item so that winning bids may be determined.

## SUMMARY OF THE INVENTION

Existing on-line auction methods are not capable of supporting the auctioning of complex items. The conventional approach is one of simple price and time comparisons for each bid on an item. In contrast, the present invention can receive a variety of data describing the subassemblies of a larger complex item and present this information in a format supporting a successful on-line auction. In response to presenting the item and subassemblies, the invention can receive and evaluate bids to determine a winner.

The present invention is generally directed to the administration of on-line auctions for complex items. Specifically, the present invention can receive information about a complex item and the subassemblies that comprise that complex item. The information can include information describing the individual subassemblies as well as data identifying how the subassemblies are interrelated and how they are arranged into larger complex items. The auction software module of the present invention is capable of taking the supplied information and creating a hierarchy that organizes the complex item and the subassemblies in a format conducive for bidding by on-line bidders. Bids for individual subassemblies or entire complex items are input by bidding clients. The auction software module receives the competing bids, evaluates them, and selects winning bids.

The hierarchy that can be created to organize the complex item and subassemblies can be viewed as similar to a family tree with many levels of generations. At the highest level, the most complex item can be labeled as a first parent. The subassemblies that comprise the first parent can be called first level descendants. Even simpler subassemblies that comprise the first level descendants can be called second level descendants and so the pattern continues until all of the subassemblies are identified. The auction software module can also verify the accuracy of the hierarchy by ensuring that no item is both an ancestor or descendant of itself.

The present invention provides a method for organizing a complex item and its subassemblies. The method for auctioning comprises presenting each complex item and its subassemblies in a graphical interface so that they can be auctioned to bidders. The graphical interface illustrates the relationships between the subassemblies and the complex item. For example, the graphical interface may present the complex item and subassemblies in an outline

format with an indentation to indicate each level of the subassemblies. The graphical interface then permits bidders to submit bids on the complex item and the subassemblies.

The present invention provides a method for an auction software module to present a complex item and its subassemblies to bidders. The auction software module presents the complex item and subassemblies to bidders in a hierarchy format so that the relationships between the complex item and subassemblies are evident. This hierarchy is stored on a server computer in a distributed computing environment so that bidders may access the hierarchy and place bids.

The present invention operates in a distributed computing environment. The invention enables the auctioning of a complex item by first receiving data about the complex item from an authoring client. An auction software module processes the data and creates a hierarchy describing the complex item and subassemblies to be auctioned. Bidding clients in the distributed computing environment can access the hierarchy of a complex item and place bids on the complex item and its subassemblies. The auction software module processes these bids and selects a winner according to the desired auction type.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A is a block diagram illustrating the operating environment for an exemplary hierarchical on-line auction system.

FIG. 1B is a functional block diagram illustrating the components of an exemplary embodiment of the present invention.

FIG. 2 is a logic flow diagram illustrating operations of a complex item administration process showing the method to define and administer a complex item structure.

FIG. 3 is a block diagram comparing the structure of a simple item, compound item, and complex item structure.

FIG. 4 is a block diagram illustrating a highly complex item structure comprising simple items, compound items, and complex items.

FIG. 5A shows a database structure for managing a highly complex item structure in accordance with an exemplary embodiment of the present invention.

FIG. 5B shows a database structure for managing bid data and relating it back to the item data structure in accordance with an exemplary embodiment of the present invention.

FIG. 6 is a logic flow diagram illustrating an exemplary process for constructing a highly complex item structure.

FIG. 7 is an illustration of a display screen depicting a hierarchical item constructor for an exemplary embodiment of the present invention.

FIG. 8 is an illustration of a display screen depicting a sub-item editor allowing a user to order quantities of a sub-item in an exemplary embodiment of the present invention.

FIG. 9 is an illustration of a display screen depicting an item hierarchy for a complex item structure in an exemplary embodiment of the present invention.

FIG. 10 is a logic flow diagram illustrating an exemplary process for generating an item hierarchy.

FIG. 11 is an illustration of a display screen depicting a hierarchical bid page for entering quantity and price bids for an item in an exemplary embodiment of the present invention.

FIG. 12 is a logic flow diagram illustrating an exemplary process for calculating the sub-assembly and total prices for a complex item to evaluate competing bids.

## DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

The present invention supports the auction of complex items in a distributed computer network. Specifically, the present invention permits an auction author connected to a server in a distributed computing environment to create an auction for a complex item comprising multiple subassemblies. Bidding clients, also connected to the server, can submit bids for the complex item and subassemblies. The invention provides a format for receiving information about the item to be auctioned and its subassemblies. The information is used to assemble a hierarchy that explains the relationships between the item and subassemblies. Furthermore, the invention presents the item and its subassemblies to bidders in a graphical user interface in such a fashion so as to facilitate bidding and selection of winners.

Although the exemplary embodiments will be generally described in the context of software modules running in a distributed computing environment, those skilled in the art will recognize that the present invention also can be implemented in conjunction with other program modules for other types of computers. In a distributed computing environment, program modules may be physically located in different local and remote memory storage devices. Execution of the program modules may occur locally in a stand-alone manner or remotely in a

client/server manner. Examples of such distributed computing environments include local area networks of an office, enterprise-wide computer networks, and the global Internet.

The detailed description which follows is represented largely in terms of processes and symbolic representations of operations in a distributed computing environment by conventional computer components, including remote file servers, remote computer servers, remote memory storage devices, one or more processing units, memory storage devices, display devices and input devices. Each of these conventional distributed computing components is accessible by the processing unit via a communications network.

The processes and operations performed by the computer include the manipulation of signals by a processing unit or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the preferred operating environment will be described.

FIG. **1A** illustrates various aspects of an exemplary distributed computing environment in which the present invention is designed to operate. Those skilled in the art will appreciate that FIG. **1A** and the associated discussion are intended to provide a brief, general description of the computer network resources in a representative computer network supporting an on-line auction.

FIG. **1A** illustrates an architecture for an exemplary embodiment of the present invention. The authoring client **140** enables an auction author to connect to a distributed computing network **105** such as the Internet. A server computer **110**, on which resides an auction software module **115**, is also connected to the network **105**. The auction software module **115** contains the instructions for conducting an auction of a complex item. Coupled to the server computer **110** is a database **120** which stores information provided by authors about complex items to be auctioned. In alternative embodiments of the invention the database **120** may be a part of the server computer **110**. Bidders are able to participate in an auction of a complex item through

bidding clients **125, 130** and **135** also connected to the network **105**. Bidding clients **125, 130,** and **135** receive information about complex items from and transmit bids to the server computer **110** via the network **105**. Although FIG. **1A** only portrays one authoring client **140** and three bidding clients **125, 130,** and **135,** the present invention can support multiple clients and conduct multiple auctions simultaneously.

Referring now to FIG. **1B**, the exemplary functions of the auction software module **115** are illustrated. FIG. **1B** depicts the auction software module functions that provide a system for defining, presenting, evaluating, and storing complex item structures in an on-line auction environment. The hierarchical item constructor **150** provides the tools for creating each item. The hierarchical item constructor **150** supports the collection, from the authoring client **140,** of the basic information and attributes that define an item in the system. The hierarchical item editor **155** is used to manage the relationships between the complex item and its subassemblies. A complex item is generally referred to as a parent, whereas the subassemblies that comprise a complex item are generally called the children. The term sibling is used to describe subassemblies of equivalent complexity that comprise the same larger complex item. This scheme of parents, children, and siblings can extend for many levels to describe a complex item comprising many subassemblies.

The hierarchical bid collector **160** provides the functionality for accepting bids that conform to the hierarchical item structure. This function manages the association of bid (price) information with each item within the item hierarchy.

The hierarchical data storage component **165** stores data for the hierarchical structures. Storage of the hierarchical item structure supports the maintenance of the information that relates one item to another.

The hierarchical item display function **170** controls how the auction information is delivered to each bidding client **125, 130,** and **135.** Hierarchical item display **170** provides the formatting technology that displays the item structure in an easy to understand format illustrating the relationships between each item (parent, sibling, child).

The hierarchical bid evaluator **175** provides the business rule validation for submitted bids according to the relationships implied by the hierarchical structure. The functions illustrated in FIG. **1B** for the auction software module **115** act in concert to reach the desired goal of selecting a winning bid in an on-line auction environment.

FIG. 2 is an overview of the hierarchical item auction process showing an exemplary method for defining and auctioning a hierarchical item structure. The exemplary process begins with the definition of each item **205**. Items are defined by the authoring client **140**, using the hierarchical item constructor **150**, by entering basic descriptive information such as name, number, description, manufacturer, and unit-of-measure. Each of these item definitions can be saved by the hierarchical data storage function **165** for later retrieval during the item hierarchy display process **170** and the bid evaluation process **175**. Once the items are defined, the process of building the item hierarchy is performed **210**. This involves identifying the relationships between parent, sibling, and child items. As the process of identifying relationships is performed, the item data structure is formed. In order to use these item hierarchies in an auction, certain parameters are defined, in step **215**, including the quantity desired for each item. Once the item hierarchy is built with the desired parameters for each item, the item hierarchy is displayed **220** to the bidding clients **125**, **130**, and **135**. In step **225**, the bidding clients **125**, **130**, and **135** submit bids for the complex item or its subassemblies to the server computer **110**. By providing a common function for displaying the hierarchical structure and collecting pricing information with a common framework, a fair environment exists for evaluating competing bids in step **230**. Finally, in step **235** the hierarchical bid evaluator **175** selects winning bid or bids according to the bid evaluation method being used for the auction.

The hierarchical approach of the present invention can support the auctioning of a wide range of possible structures. Using block diagrams, FIG. 3 and FIG. 4 illustrate exemplary structures of item assemblies which can be auctioned with the hierarchical architecture of the present invention. A simple item is a single item **305** with no children or siblings. The simple item **305** requires no relationships to be defined. A compound item **310** is a simple item assembly that has child items. A complex item **315** is an item assembly that has both child items and child item assemblies, also called subassemblies. A highly complex item **405** is an item assembly with multiple levels of child complex items (subassemblies). FIG. 4 represents a complex item **405** with 4 levels of subassemblies. Item Assembly 1 has a Sub Assembly 1D, which has a Sub Assembly 1D2, which has a Sub Assembly 1D2A, which has a Sub Assembly 1D2A2.

Referring to FIG. 5A and FIG. 5B, the storage of the hierarchical information typically requires a relational database management system such as Microsoft SQL Server. This structure

allows the hierarchical information to be represented in tables that have relational keys allowing the parent/child relationship to be enforced. FIG. 5A shows an exemplary table structure for defining complex item data structures. The `tItem` table **505** provides the storage for the descriptive information about each item while the `tItemBundle` table **510** provides the storage for the information that relates items together. The `ParentItemID` field of the `tItemBundle` table and the `ItemID` field of the `tItemBundle` table each relate to the `tItem` table and thus provide the mechanism for relating items together in a parent/child relationship. The `tItemBundle` table **510** contains information for defining quantity and display sort order. This information is specific to an individual relationship between items. If Item 1A is related to Item 1 as a child, then the quantity of Item 1A as it relates to Item 1 is information that should be maintained. In addition, for display purposes, it is important to allow the user to control the display sequence of siblings that are related together. The display sort order field is where this information is maintained.

Once an item hierarchy is presented to a bidding client **125**, the hierarchical bid evaluator **175** will process the bid data. The hierarchical data storage function **165** can record the bid data in a table such as the `tBid` table **515** and the `tBidData` table **520** shown in FIG. 5B. The `tBid` table **515** comprises fields that identify each and every distinct bid in a summary format. This includes information that identifies the specific auction (`OfferingID`) the bid is for, the identity of the bidder (`BidderID`), the time submitted (`SubmitTime`), the total price of the bid (`SubmitPrice`), the status of the bid whether winning or losing (`BidStatusType`), the price assigned to the bid by the system if the evaluation algorithm alters the bid (`AssignedPrice`), a flag indicating if this bid was placed via proxy (`IsProxy`), an optional comment entered by the bidder (`Comment`), object storage for the `tBidData` record that represents all the bid detail (`BidData`), the expiration date/time for the bid (`ExpireTime`), and the update time of the bid (`LastUpdateTime`).

The bid data comprises the bid details and is represented in FIG. 5B as the `tBidData` table **520**. It provides the information necessary to relate price and quantity information to the specific item for which it was entered. This is done through `ItemID` and `OfferingItemID` fields. These provide specific identities for each of the items within the complex item structure.

The `OfferingItemID` identifies the top-level (parent) item for which the bid was placed while the `ItemID` identifies the sub-item (child) for which the bid was entered. The `ItemQuantity` field is the original quantity defined for the sub-item while the `BidQuantity` field comprises the quantity entered by the bidder. Bid quantity can be equal to or less than the `ItemQuantity` value. The `BidPrice` field contains the price entered by the user for the sub-item. The `RollupPrice` field includes a value for the calculated price of the total of all the sub-items and is generally only present for complex assemblies. The `Children` and `ChildStatus` fields are used in the display construction process, as discussed below in reference to FIG. 10.

The hierarchical item constructor **150** and hierarchical item editor **155** support the grouping of items together to form subassemblies and the grouping of subassemblies to form complex assemblies. The grouping of items and subassemblies into complex assemblies is how the hierarchies that are presented to bidders are created. Two sub-items attached to a parent are both children of the parent as well as siblings of each other. During the hierarchy construction, additional information is collected that is relevant only to the specific relationship, such as the quantity of the sub-item attached and the display sort order of the sub-item. The information defining the items and item hierarchy is managed by the hierarchical data storage function **165** so that it may be retrieved for display.

Items may exist more than once within the item structure. The same item may exist multiple times at the same level or at different levels. However, an item may not exist as a descendant of itself. This creates a recursion problem. Consequently, the process of creating the item structure includes the appropriate verification of the item structure. When an item is added to the item hierarchy, the verification process checks to ensure that the new item will not cause a recursion problem within the item hierarchy. FIG. 6 is a logic flow diagram of an exemplary verification process. The verification process involves stepping through the entire item structure, examining each item with a consistent set of tests as follows:

1) Does an item exist as a descendent of itself? This test involves 'walking down' the structure.

2) Does an item exist as an ancestor of itself? This test involves 'walking up' the structure. This test is basically the same as the previous test but is necessary within the context of the process to ensure that the entire structure is evaluated.

Referring to FIG. 6, the following steps are taken to eliminate recursion problems. The first step is to determine if there are identical child and parent items. In step **605**, the item to be added is compared to a parent item. If the two are the same, the item is rejected in step **608**. If they are not the same, the "No" branch is followed to step **610**. In step **610**, the item to be added becomes the parent and the auction software module **115** retrieves a list of child items for the item to be added. In step **615**, the auction software module **115** queries the `tItem` data structure **505** to see if the child item is the same as the parent item. If this is the case, the "Yes" branch is followed and the item is rejected in step **620**. If this is not the case, the "No" branch is followed to step **625** and the examination of the structure continues with any existing children of the child item. If there are existing children, the "Yes" branch is followed back to step **610** and the examination is repeated. The verification process continues until the base of the complex item structure is reached.

If there are no existing children, the process of "walking down" the structure is completed and the "No" branch is followed to step **630** where the evaluation continues by determining if the item is a child and an ancestor of itself. This begins by retrieving a list of parents of the parent item in step **630**. The parent items are examined to determine if the parent is the same as the child in step **635**. If there is a match, the "Yes" branch is followed to step **640** and the item is rejected. If the parent is not the same as the child, the "No" branch is followed to step **645** and the process continues with the examination of any parents of the parent item. If there are parents, the "Yes" branch is followed back to step **630** and the process is repeated. If there are no additional parents, the "No" branch is followed to step **650** and the item is added.

The process of defining this structure in an on-line environment presents many challenges. The primary goal is to provide a user interface that is easy to use and understand, while providing sufficient information and flexibility to allow the user to construct the necessary item structures. The integral steps in conducting an on-line auction are defining each item and linking the items together in the appropriate relationship. These steps can be supported by the use of display screens for defining the items and for relating the items together.

FIG. 7 depicts an exemplary graphical user interface **705** for defining an item. Separate fields are shown for the entry of Item Name, Item Number, Description, Additional Information, Manufacturer, Manufacturer Item Number, and Unit of Measure. This screen shows a list of exemplary sub-items that have been related to this item. It shows Item 1A, Sub Assembly 1B, Sub Assembly 1C, and Sub Assembly 1D as being sub-items. The auction author may add additional sub-items by clicking the Add New Sub-item button or may edit the sub-item attributes (i.e., quantity and display sort order) by clicking the Update Sub-items button. In addition, the auction author may access the complex item hierarchy by clicking the View Item Hierarchy button. As sub-items are added, they are displayed in the sub-item list in the order as defined by their display sort order. The quantities assigned are displayed in the quantity column. The auction author may delete a sub-item by clicking the Delete button in the Action column. Deleting a sub-item will not remove the sub-item from the system but will simply delete the relationship of the sub-item to the parent item.

FIG. 8 depicts an exemplary screen of a graphical user interface **805** for updating information about each sub-item associated to the parent item. This involves the entry of the display sort order and the item quantity. This screen displays each sibling item that is associated to the parent and provides fields for entering the display sort order and the quantity. The item name, item number and units of measure are provided for information. A Delete button is also provided that enables the auction author to remove a sub-item from the list. Once the auction author has made all the changes to sort order and quantity, she will click the Update Sub-items button to record the changes.

FIG. 9 illustrates an exemplary screen of a graphical user interface **905** for displaying the item hierarchy in the hierarchical display format. This format enables the auction author or

in FIG. 9 are buttons to add a new sub-item and to update the sub-items, which can be done at a display screen such as the one shown in FIG. 8.

FIG. 9 shows within the display **905** the fully expanded hierarchy of a representative complex item structure called Item Assembly 1. It shows that Item Assembly 1 has 4 children related that are Item 1A, Sub-Assembly 1B, Sub-Assembly 1C, and Sub-Assembly 1D. Sub-Assembly 1B is represented as a compound item with 2 children that are Item 1B1 and Item 1B2. Sub-Assembly 1C is represented as a compound item with 5 children that are Item 1C1, Item 1C2, Item 1C3, Item 1C4, and Item 1C5. Sub-Assembly 1D is represented as a complex item structure with 2 children that are Sub-Assembly 1D2A and Item 1D2B. Sub-Assembly 1D2A is represented as a complex item structure with 2 children that are Item 1D2A1 and Sub-Assembly 1D2A2. Sub-Assembly 1D2A2 is represented as a compound item with 2 children that are Item 1D2A2A and Item 1D2A2B. The invention also supports additional levels of sub-assemblies.

As described in FIG. 2, once an auction author defines the items to be auctioned in step **205**, the auction software module **115** builds the item hierarchy in step **210**. A logic flow diagram illustrating, in greater detail, an exemplary process for constructing an item hierarchy is represented in FIG. 10. The exemplary process of FIG. 10 employs a scheme using indentation and graphical icons to represent a complex item. Alternative embodiments of the present invention may employ other graphical layouts, such as two-dimensional charts, circular charts, and three-dimensional pictorials, to represent a complex item.

The process defined in FIG. 10 begins by retrieving the hierarchical recordset in step **1005**. Each recordset row contains three columns used for rendering the user interface. These columns are Children, IndentLevel, and ChildStatus. Children represents the number of children in a row. IndentLevel is a range of numbers from 1 to n for the indention level of the row. ChildStatus is 1 for the first child, 0 for a middle child, 2 for the last child and 3 if the item is an only child. In step **1010**, the auction software module **115** determines the row indent level for an item. If the indent level is different from the last indent level in step **1015**, the "Yes" branch is followed to step **1020** where the indent level is corrected. If the indent level is not different from the last indent level, the flow chart proceeds directly to step **1025** where the auction software module **115** identifies whether the current row is a root row. A root is the highest level of complexity for an item and has an indent level of 1. If the row is not a root row, the row contains a child and the "No" branch is followed to step **1030** where the appropriate indentation and

symbol are inserted into the hierarchy to represent a child row. If the row is a root row, the row does not need to be indented and the "Yes" branch is followed to step **1035** where the appropriate symbol for a root row is inserted into the hierarchy.

In step **1040** the item name and quantity are inserted into the row in the hierarchy. The auction software module is now ready to move onto the next item which will be inserted into the next row in the hierarchy. In step **1045**, the auction software module **115** queries the `tItem` data structure **505** to ascertain whether the row that was just completed has children. If the row does have children, the "Yes" branch is followed to step **1050** where the hierarchy is adjusted to create a next row of children descending from the row just completed. If the row does not have children, the "No" branch is followed and the flow chart proceeds directly to step **1055** where the indent level is saved as the last indent level. The last indent level is saved so it can be used as a reference point for starting the next row. In step **1060**, the auction software module **115** queries whether there is another row to create. If so, the "Yes" branch is followed and returns to step **1010** where the indent level for the next row is determined. If there are no remaining rows to be created, the hierarchy is complete.

With the item structure complete and stored in the database **120**, it is available for use in an on-line auction environment. The on-line auction environment involves many bidding clients **125**, **130**, and **135** entering pricing information for each element within the item structure. This involves the process of displaying the hierarchical item structure to the bidder in a format that will show the relationships between all the items and allow the bidder to enter bid quantity and bid price information for each item in the structure.

FIG. 11 represents a screen from an exemplary graphical user interface **1105** for displaying the item hierarchy and collecting the quantity and price information for each item. This screen provides pertinent information related to the specific auction being conducted at the top of the screen. This includes information that the bidder will require in order to place an appropriate bid such as Starting Price, Bid Increment, Low Bid Price, Reserve Price, Time Remaining, Status, and Bid Count. The Bid Information displays the items within the hierarchy. The Bid Information display constructs the item hierarchy according to the process described in FIG. 10.

The bidder enters the price and quantity information associated with each item in the structure. In one embodiment of the present invention, price and quantity cannot be entered for

subassemblies as this information is automatically calculated from bids for larger assemblies. In another embodiment of the present invention, bids may be made directly on subassembly items. The price and quantity information can be processed to calculate the sub-total for each and every sub-assembly in the structure. A total bid can also be calculated by adding the total price for each of the children for the parent item.

FIG. 12 shows the process for calculating the price for each sub-assembly and the total bid price so that competing bids can be evaluated. It begins by retrieving the bid data from the `tBid` data table **515** in step **1205**. The page is initialized with arrays of row data based on the item hierarchy structure. All subtotals are initialized to zero in step **1210**. In step **1215**, the auction software module **115** queries whether the current row corresponds to the correct indent level. If there is no correspondence, the "No" branch is followed to step **1220** where the auction software module **115** looks to the next row for a match. This step is repeated until a match is found and, in step **1225**, the auction software module **115** verifies whether the bidder's entry is valid. If the bidder has entered data in the wrong row or the price and quantity data is not in the correct format, a validation error will be displayed in step **1230** and the auction software module will start over by moving to the next row in step **1220**.

Once the bidder's input is validated, the auction software module **115** computes the row subtotal in step **1235**. In step **1240**, the auction software module queries for children. If there are children of this row, those prices will be added into the row subtotal in step **1245**. Otherwise, in step **1250**, the auction software module **115** looks to see if the row is a child. If the row is a child, its subtotal must be added to a preceding parent row in step **1255**. The auction software module **115** then proceeds to step **1260**, to examine any remaining rows. If there are remaining rows, the auction software module **115** returns to step **1215** to begin working on another row. If there are no remaining rows, the subtotals and grand totals are displayed, as shown in FIG. 11, for this particular bidding client in step **1265**. In step **1270**, the auction software module **115** repeats the entire process for any other bidding clients. Once the subtotals and totals are computed for all bidding clients, the auction software module can compare the totals and select the winning bids. Winning bids will be determined by several factors including the particular auction type and to what level of subassembly separate bidding is permitted. These factors may vary from auction to auction.

In summary, the present invention enables and supports the auctioning of complex items in a distributed computing environment. The invention allows an auction author to submit information about a complex item and its subassemblies that are to be auctioned to multiple bidders. The invention can present information about the complex item and its subassemblies in a manner suitable for an on-line auction. Finally, the invention can accept bids on the complex item and its subassemblies from multiple bidders and determine the winning bid or bids.

It will be appreciated that the present invention fulfills the needs of the prior art described herein and meets the above-stated objects. While there has been shown and described the preferred embodiment of the invention, it will be evident to those skilled in the art that various modifications and changes may be made thereto without departing from the spirit and the scope of the invention as set forth in the appended claims and equivalence thereof. For example, the present invention can support the auctioning of items other than complex goods such as complex services or contract options. The invention can also be used in commercial contexts other than an auction such as the sale of inventory or bidding on contracts where there are multiple components.

## CLAIMS

What is claimed is:

1. A method for administering complex item structures for auctions in a distributed computing environment, comprising:

defining a complex item and its subassemblies that are to be auctioned to bidders, wherein a complex item comprises several levels of subassemblies;

building a hierarchy for the complex item and its subassemblies based on relationships between the item and its subassemblies;

accessing the hierarchy of the complex item and its subassemblies on a server computer by bidders coupled to the server computer to view the relationships between the complex item and its subassemblies;

sending competing bids for the complex item and its subassemblies from the bidders to the server computer; and

selecting winning bids for the complex item and its subassemblies.

2. The method of claim 1, wherein the step of defining a complex item and its subassemblies further comprises:

entering descriptive information for each item and subassembly.

3. The method of claim 1, wherein the step of building a hierarchy for the item and its subassemblies comprises:

identifying a first item and representing it as a first parent;

identifying a subassembly of the first parent, if any, and representing it as a first level descendant;

identifying a subassembly of the first level descendant, if any, and representing it as a second level descendant;

identifying another subassembly of the first parent, if any, and representing it as a sibling to the first level descendant; and

repeating the foregoing steps until there are no remaining items and subassemblies to be identified.

4. The method of claim 3, wherein the step of building a hierarchy for the complex item and its subassemblies further comprises:

comparing each item and its subassemblies to their respective descendants, and

if one of the descendants is the same as one of the ancestors, then deleting the descendant from the hierarchy.

5. The method of claim 1, wherein the step of accessing the hierarchy of the complex item and its subassemblies by bidders connected to the server computer comprises:

viewing a graphical interface operative to

present the hierarchy of the complex item and its subassemblies to show relationships between the complex item and its subassemblies;

accept a bid on the item and its subassemblies; and

display the time remaining for bidding and the status of bids.

6. The method of claim 1, wherein the step of sending competing bids for the complex item and its subassemblies from the bidders to the server computer comprises:

entering a bid quantity and

entering a bid price.

7. The method of claim 1 wherein the step of selecting winning bids for the item and its subassemblies on the server computer further comprises:

calculating a bid price for each subassembly;

calculating a total bid price;

recording a total bid price, a bidder identity, and a bidding time; and

evaluating the total bid price to determine a winner of the auction.

8. A computer-readable medium having computer-executable instructions for performing the steps recited in claim 1.

9. A method for organizing a complex item and its subassemblies into a format for auctioning comprising:

presenting a complex item to be auctioned in a graphical interface;

presenting subassemblies of the complex item to be auctioned in the graphical interface; and

presenting the relationship between the complex item and the subassemblies in the graphical interface.

10. The method of claim 9 wherein the relationships between the complex item and the subassemblies are identified by presenting each level of descendants in an outline format, each descendant level designated by an indented level.

11. The method of claim 9 wherein the graphical interface is operable for:

enabling bidders to bid on the complex item and its subassemblies by showing the relationships between the complex item and its subassemblies,

accepting bids on the complex item and its subassemblies, and

displaying the time remaining for bidding and the status of bids.

12. A computer-readable medium having computer-executable instructions for performing the steps recited in claim 9.

13. A computer system capable of supporting an auction of a complex item and its subassemblies, wherein a complex item comprises several subassemblies, comprising:

a server computer logically connected to two or more clients;

a software module residing on the server computer and capable of receiving and providing information on a complex item and its subassemblies in a format for supporting an auction;

an authoring client logically connected to the server computer that provides information to the software module about the complex item and its subassemblies; and

one or more bidding clients logically connected to the server computer and capable of receiving information from the software module about the complex item and the subassemblies being auctioned and providing bids on the complex item and the subassemblies.

14. The system of claim 13, wherein the software module residing on the server is further operable for:

receiving information from the authoring client describing the complex item and its subassemblies;

grouping the subassemblies of the complex item in a format so that bidding clients may bid on them;

receiving bids from one or more bidding clients for the complex item and the subassemblies; and

selecting winning bids for the complex item and its subassemblies.

15. The system of claim 13, wherein the software module residing on the server is further operable for building a hierarchy for the complex item and its subassemblies by

identifying a first complex item and representing it as a first parent;

identifying a first subassembly of the first parent, if any, and representing it as a first level descendant;

identifying a second subassembly of the first level descendant, if any, and representing it as a second level descendant;

identifying a third subassembly of the first parent, if any, and representing it as a sibling to the first level descendant; and

repeating the foregoing steps until there are no remaining items and subassemblies to be identified.

16. The system of claim 13, wherein the software module residing on the server is further operable for

comparing each complex item and its subassemblies to their respective descendants, and

if one of the descendants is the same as one of the ancestors, then deleting the descendant from the hierarchy.

17. The system of claim 13, wherein the software module residing on the server is further operable for displaying a graphical interface operative to

present the hierarchy of the complex item and its subassemblies to show relationships between the complex item and its subassemblies;

accept a bid on the complex item and its subassemblies; and

display the time remaining for bidding and the status of bids.

18. The system of claim 13, wherein the software module residing on the server is further operable for

calculating a bid price for each subassembly;

calculating a total bid price;

recording a total bid price, a bidder identity, and a bidding time; and

evaluating the total bid price to determine a winner of the auction.

19. A method for presenting a complex item and its subassemblies to bidders in an auction comprising:

presenting a hierarchy of a complex item and its subassemblies that shows the relationships between the complex item and its subassemblies;

accepting bids from bidders on the complex item and its subassemblies; and

displaying the time remaining for bidding and the status of bids.

20. The method of claim 19 further comprising

presenting the hierarchy with a graphical user interface created by a server computer; and

accepting the bids from bidders operating clients coupled to the server computer.

ARCHITECTURE

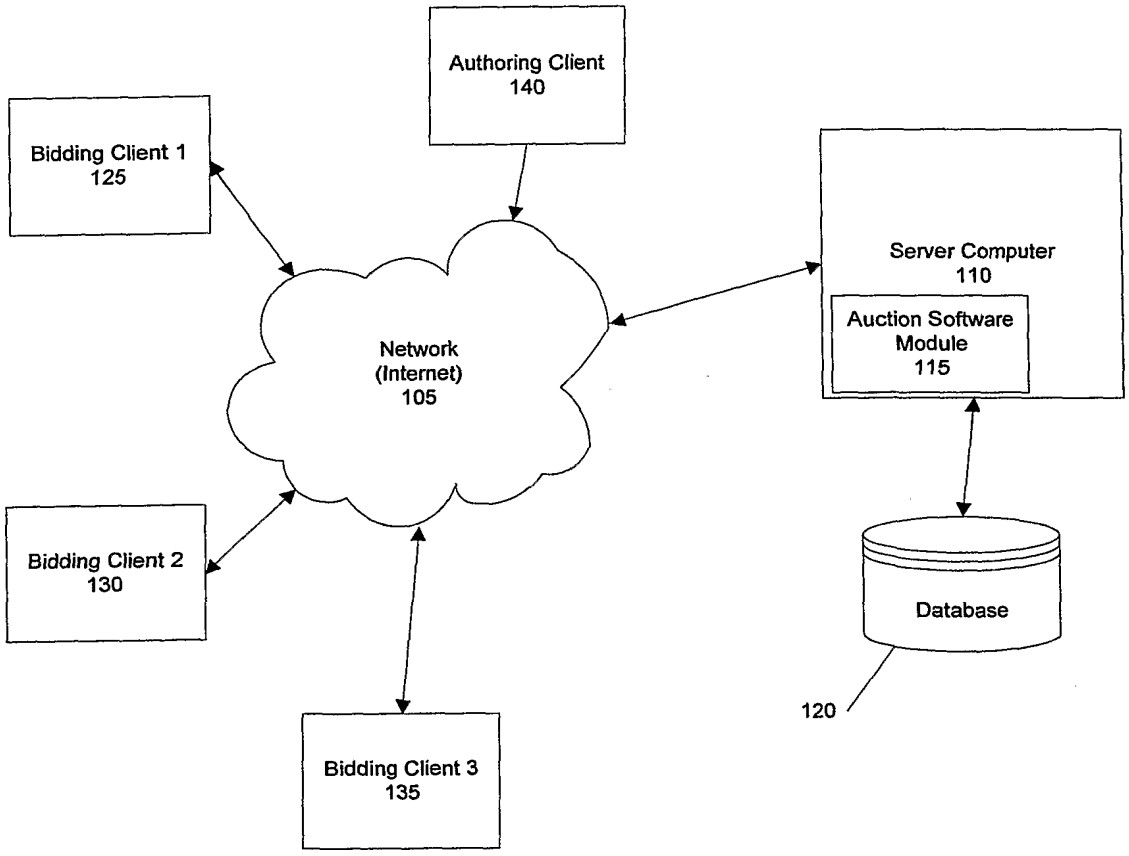



FIG. 1A

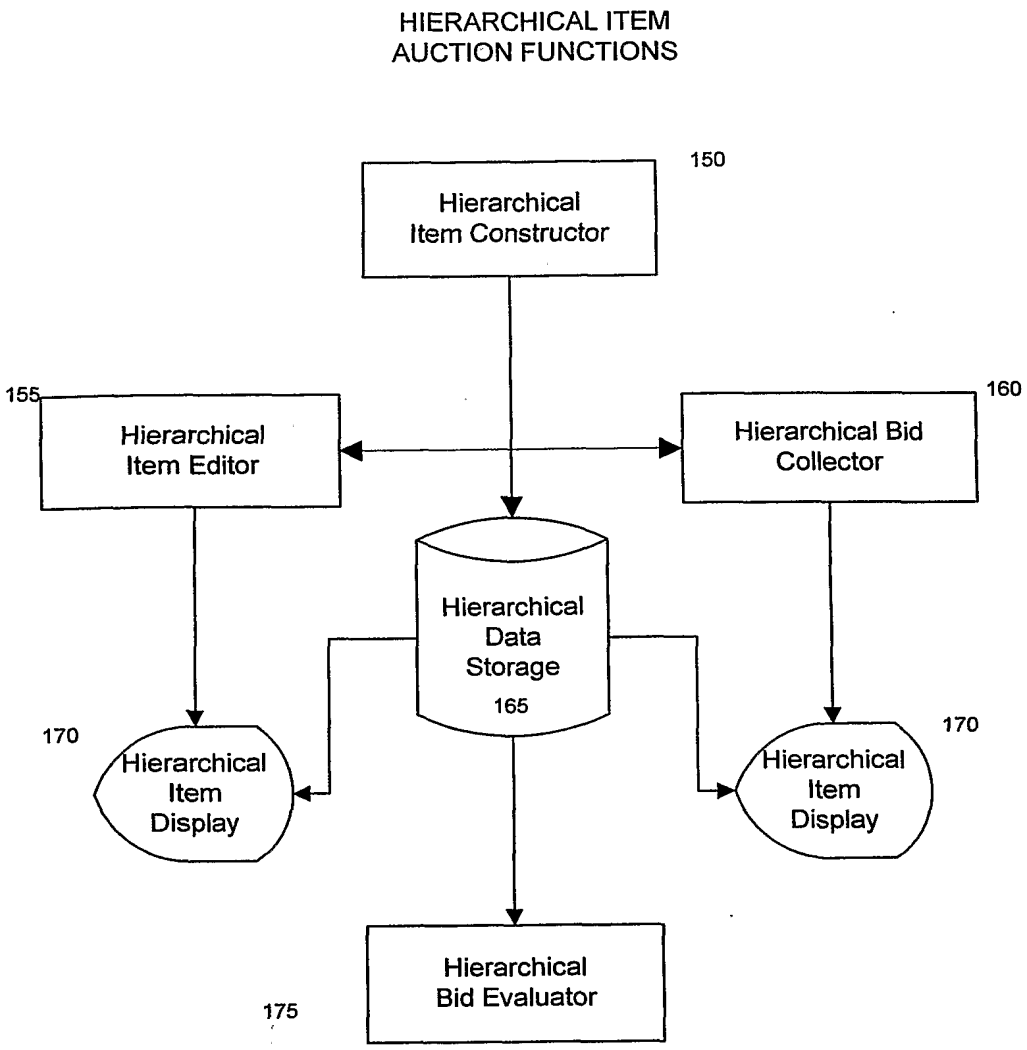

HIERARCHICAL ITEM
AUCTION FUNCTIONS
Hierarchical
Item Constructor
150
155
Hierarchical
Item Editor
160
Hierarchical Bid
Collector
Hierarchical
Data
Storage
165
170
Hierarchical
Item
Display
170
Hierarchical
Item
Display
Hierarchical
Bid Evaluator
175

FIG. 1B

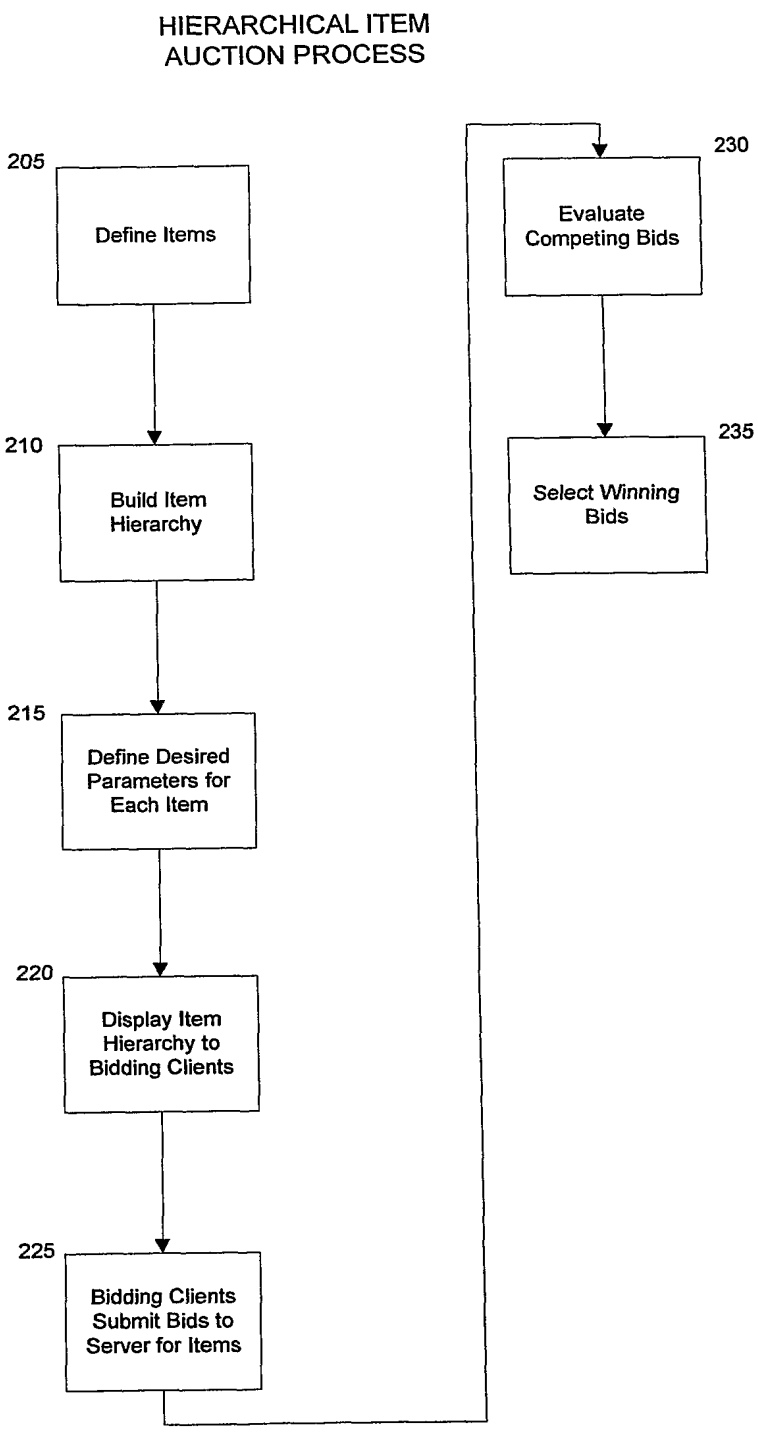

HIERARCHICAL ITEM AUCTION PROCESS
205
Define Items
210
Build Item Hierarchy
215
Define Desired Parameters for Each Item
220
Display Item Hierarchy to Bidding Clients
225
Bidding Clients Submit Bids to Server for Items
230
Evaluate Competing Bids
235
Select Winning Bids

FIG. 2

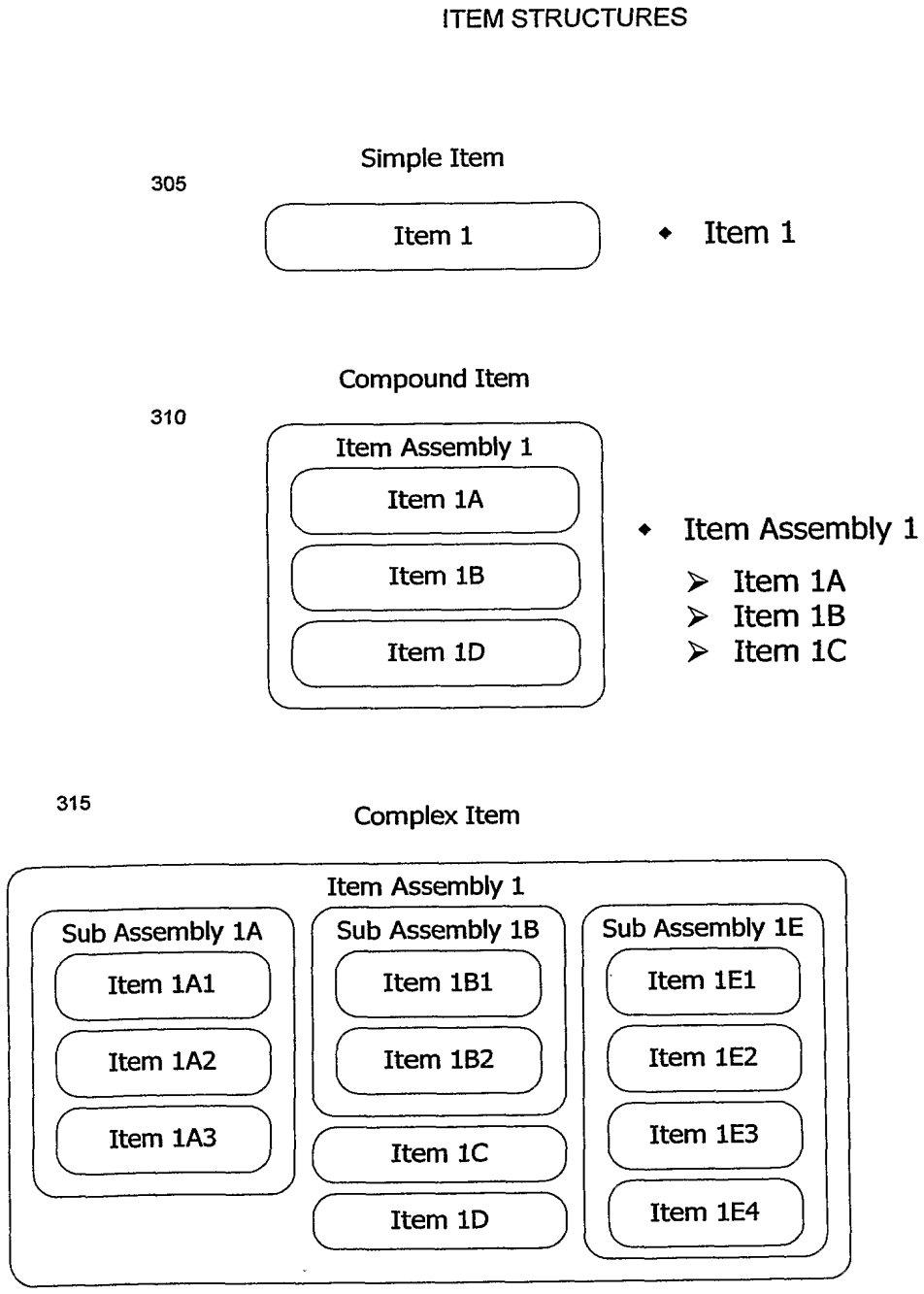

ITEM STRUCTURES
Simple Item
305
Item 1
Item 1
Compound Item
310
Item Assembly 1
Item 1A
Item 1B
Item 1D
Item Assembly 1
Item 1A
Item 1B
Item 1C
315
Complex Item
Item Assembly 1
Sub Assembly 1A
Item 1A1
Item 1A2
Item 1A3
Sub Assembly 1B
Item 1B1
Item 1B2
Item 1C
Item 1D
Sub Assembly 1E
Item 1E1
Item 1E2
Item 1E3
Item 1E4

FIG. 3

ITEM STRUCTURES

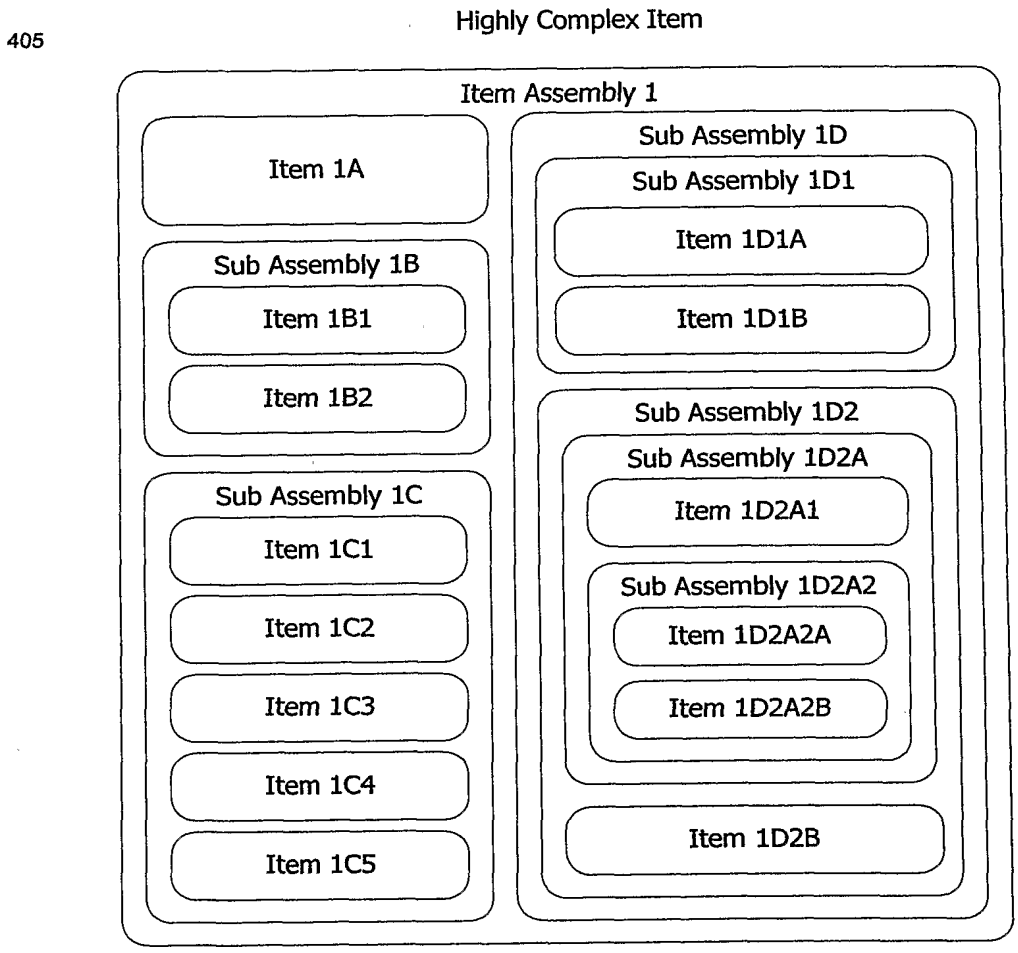



FIG. 4

COMPLEX ITEM DATA
STRUCTURE

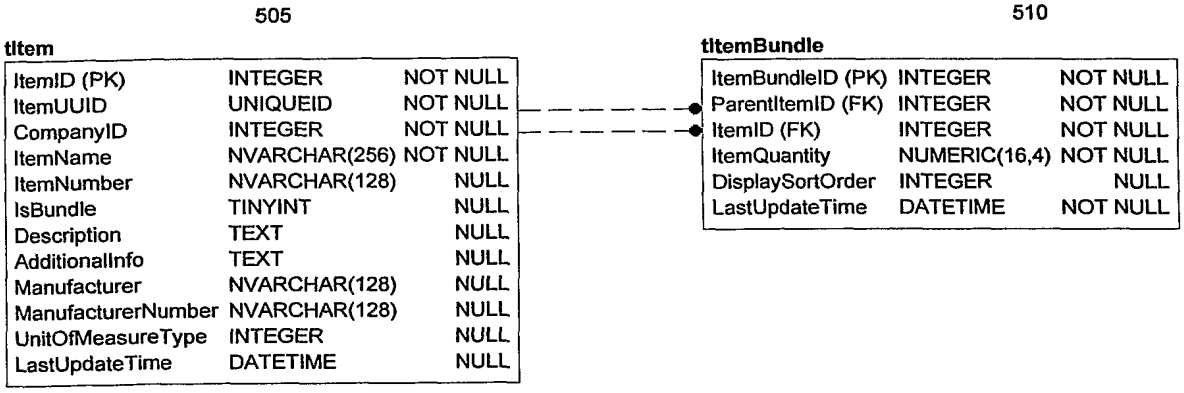



FIG. 5A

BID DATA STRUCTURE

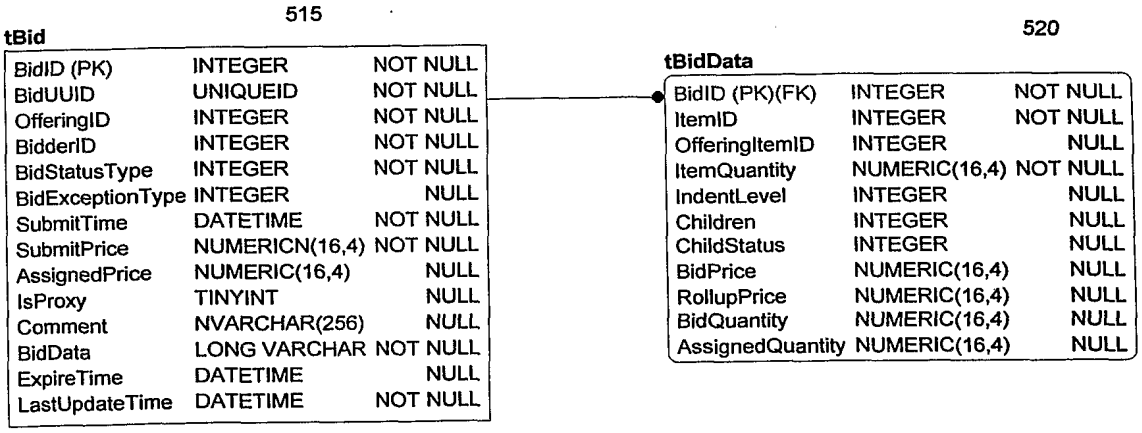



FIG. 5B

VERIFICATION PROCESS

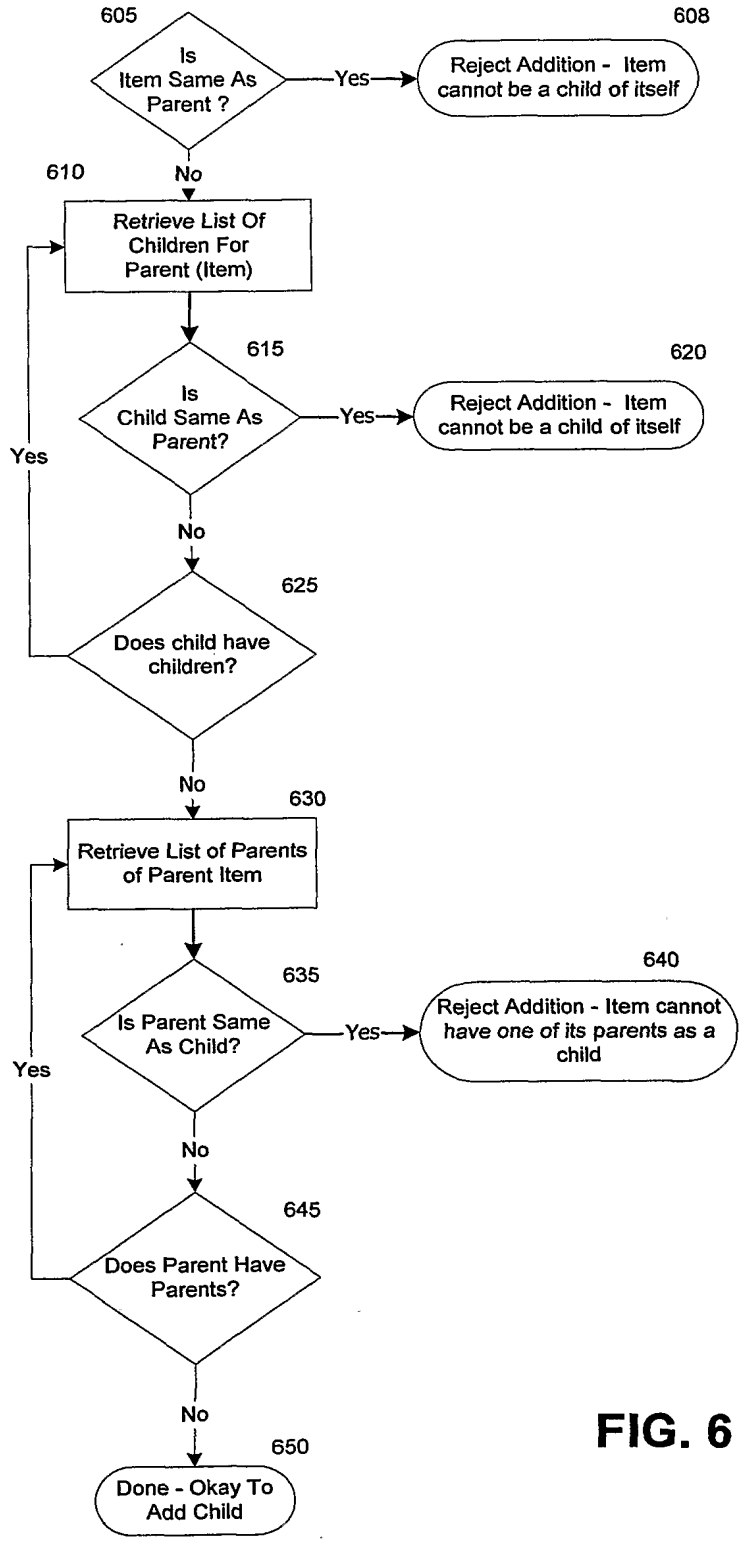



FIG. 6

ITEM UPDATE SCREEN

705

Procuri.com - Buyers Resource Center - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.procuri.com/Admin/Items/EditItem.asp

## Item Administration - Update Item

| Item Information (* = Required Entry) | |
|---|---|
| Item Name | Item Assembly 1 |
| Item Number | 1 |
| Description | |
| Additional Info | |
| Manufacturer | |
| Manufacturer Item Number | |
| Unit Of Measure | - No Unit Of Measure - |

**Subitems**

| Name | Number | Quantity | Units | Action |
|---|---|---|---|---|
| Item 1A | 1A | 1 | | Delete |
| Sub Assembly 1B | 1B | 1 | | Delete |
| Sub Assembly 1C | 1C | 1 | | Delete |
| Sub Assembly 1D | 1D | 1 | | Delete |

Add New Subitems | Update Subitems | View Item Hierarchy

**Product Categories**

| | |
|---|---|
| Private Categories | ☑ Private Category 1<br>☐ Private Category 2<br>☑ Private Category 3 |
| Public Categories | ☐ Public Category 1<br>☑ Public Category 2<br>☑ Public Category 3 |

Done Internet

FIG. 7

SUB-ITEM UPDATE SCREEN

805

Procuri.com - Buyers Resource Center - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.procuri.com/Admin/Items/EditSubItems.asp

## Item Administration - Update Subitems

| Parent Item Information | | | | | |
|---|---|---|---|---|---|
| **Item Name** | Item Assembly 1 | | | | |
| **Item Number** | 1 | | | | |
| **Subitem Name** | **Subitem Number** | **Sort** | **Quantity** | **Units** | **Action** |
| Item 1A | 1A | 1 | 1 | | Delete |
| Sub-Assembly 1B | 1B | 2 | 1 | | Delete |
| Sub-Assembly 1C | 1C | 3 | 1 | | Delete |
| Sub-Assembly 1D | 1D | 4 | 1 | | Delete |

Update Subitems Reset

Return To Item Detail

Done Internet

FIG. 8

ITEM HIERARCHY SCREEN

905

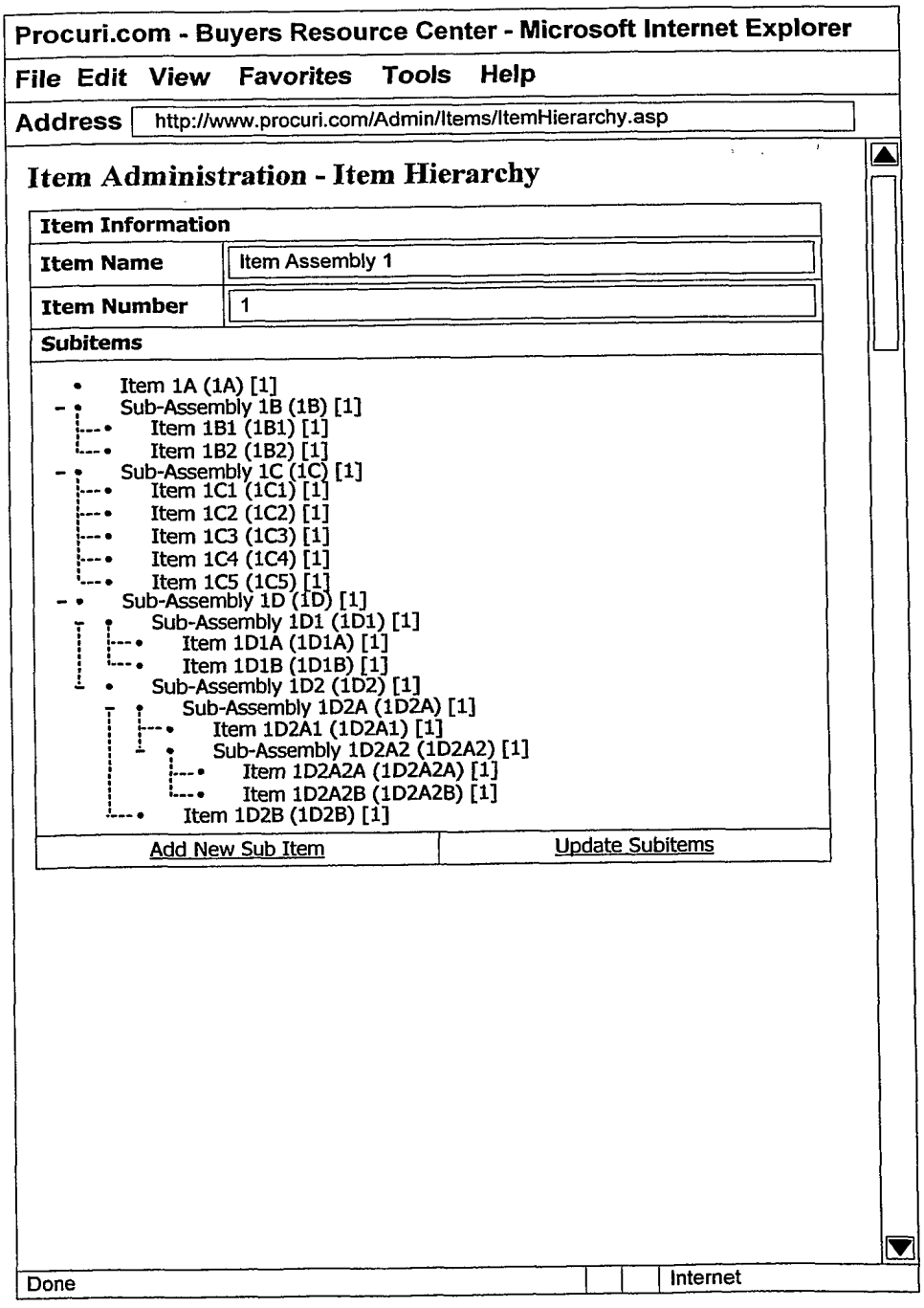



FIG. 9

BUILDING ITEM HIERARCHY

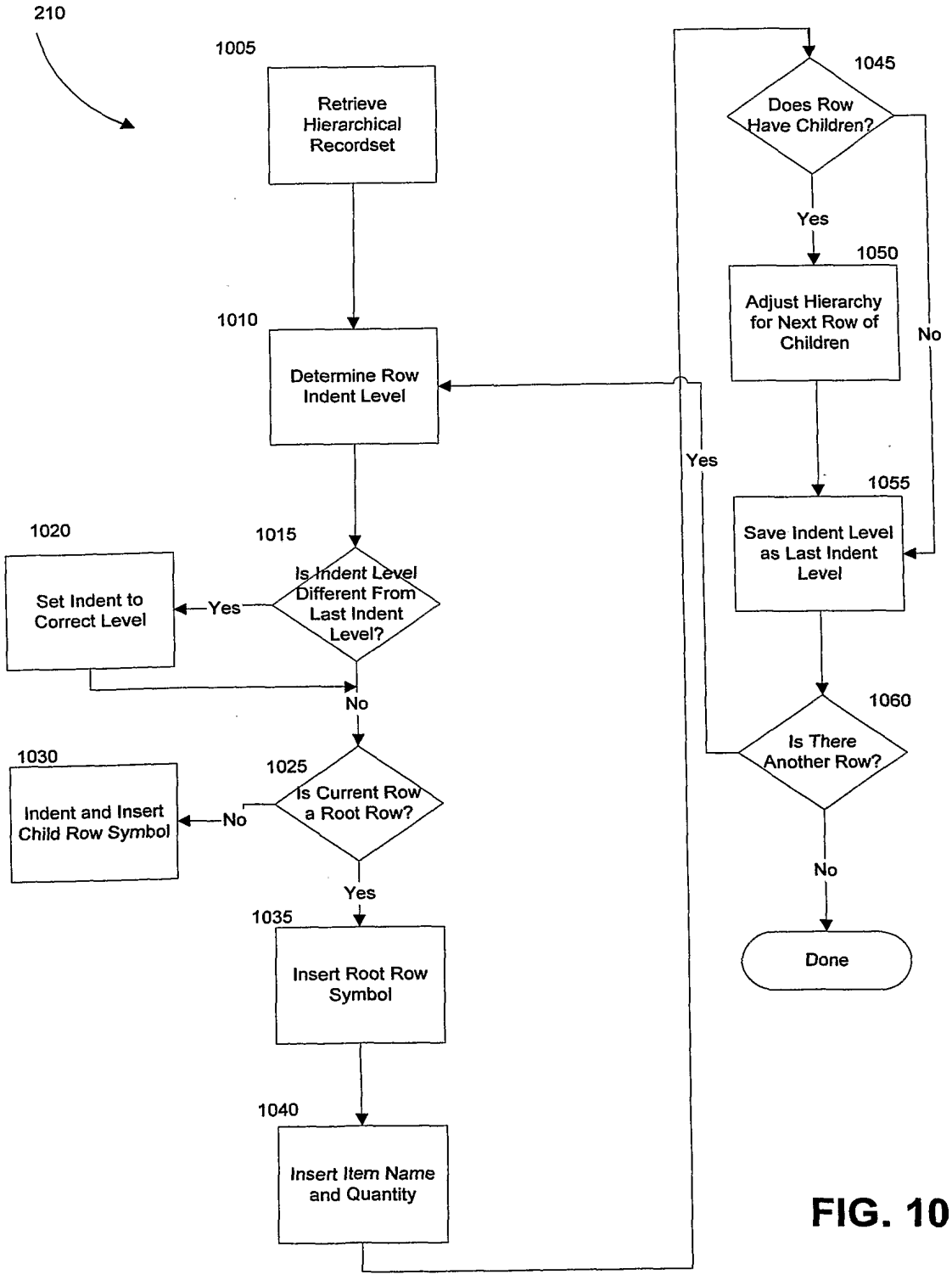



FIG. 10

BID SUBMIT SCREEN

1105

Procuri.com - Suppliers Resource Center - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.procuri.com/Bidding/Bid.asp

## Submit Bid - Acme Supply Company

| RFQ Title | Item Assembly 1 Acquisition | | |
|---|---|---|---|
| Starting Price | $100,00.00 USD | Time Remaining (h:m:s) | 95:51:03 |
| Bid Increment | $0.01 USD | RFQ Status | OPEN |
| Low Bid Price | No Bids | Bid Count | 0 |
| Reserve Price | Not Available | Reserve Price Met | NO |

**Bid Information**

| Description | Units | Qty | Price | Sub Total |
|---|---|---|---|---|
| Item Assembly 1 | n/a | 1 | 8625 | 8625 |
| Item 1A | n/a | 1 | 100 | 100 |
| Sub-Assembly 1B | n/a | 1 | 125 | 125 |
| Item 1B1 | n/a | 1 | 50 | 50 |
| Item 1B2 | n/a | 1 | 75 | 75 |
| Sub-Assembly 1C | n/a | 1 | 150 | 150 |
| Item 1C1 | n/a | 1 | 10 | 10 |
| Item 1C2 | n/a | 1 | 20 | 20 |
| Item 1C3 | n/a | 1 | 30 | 30 |
| Item 1C4 | n/a | 1 | 40 | 40 |
| Item 1C5 | n/a | 1 | 50 | 50 |
| Sub-Assembly 1D | n/a | 1 | 8250 | 8250 |
| Sub-Assembly 1D1 | n/a | 1 | 500 | 500 |
| Item 1D1A | n/a | 1 | 250 | 250 |
| Item 1D1B | n/a | 1 | 250 | 250 |
| Sub-Assembly 1D2 | n/a | 1 | 7750 | 7750 |
| Sub-Assembly 1D2A | n/a | 1 | 2750 | 2750 |
| Item 1D2A1 | n/a | 1 | 500 | 500 |
| Sub-Assembly 1D2A2 | n/a | 1 | 2250 | 2250 |
| Item 1D2A2A | n/a | 1 | 1000 | 1000 |
| Item 1D2A2B | n/a | 1 | 1250 | 1250 |
| Item 1D2B | n/a | 1 | 5000 | 5000 |
| **Total Bid** | | | | 8625 |

**Comment**

Calculate Submit Bid Reset

Done Internet

FIG. 11

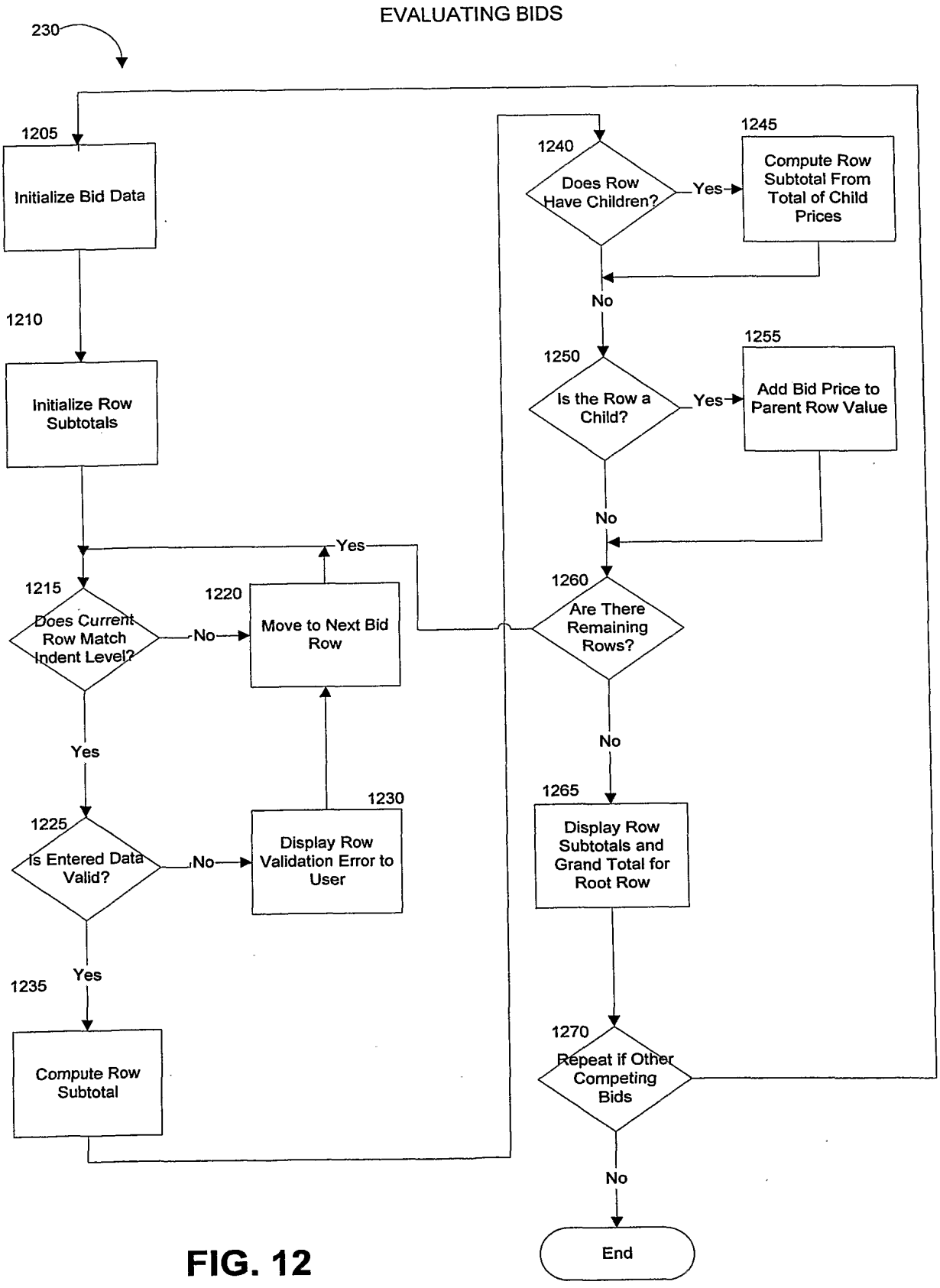

EVALUATING BIDS
230
1205
Initialize Bid Data
1210
Initialize Row Subtotals
1215
Does Current Row Match Indent Level?
No
1220
Move to Next Bid Row
Yes
1225
Is Entered Data Valid?
No
1230
Display Row Validation Error to User
Yes
1235
Compute Row Subtotal
1240
Does Row Have Children?
Yes
1245
Compute Row Subtotal From Total of Child Prices
No
1250
Is the Row a Child?
Yes
1255
Add Bid Price to Parent Row Value
No
1260
Are There Remaining Rows?
No
1265
Display Row Subtotals and Grand Total for Root Row
1270
Repeat if Other Competing Bids
No
End

FIG. 12